Importing the Dependencies

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.datasets
from sklearn.model_selection import train_test_split
```

Data Collection & Processing

```
breast_cancer_dataset = sklearn.datasets.load_breast_cancer()

print(breast_cancer_dataset)
```

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01,
4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]]), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0,
0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0,
1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0,
0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
```

```
1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0,
0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1,
1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0,
0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
'frame': None, 'target_names': array(['malignant', 'benign'],
dtype='<U9'), 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer
wisconsin (diagnostic) dataset\
n--------------------------------------------\n\n**Data Set
Characteristics:**\n\n    :Number of Instances: 569\n\n    :Number of
Attributes: 30 numeric, predictive attributes and the class\n\
n    :Attribute Information:\n        - radius (mean of distances from
center to points on the perimeter)\n        - texture (standard
deviation of gray-scale values)\n        - perimeter\n        - area\n
- smoothness (local variation in radius lengths)\n        -
compactness (perimeter^2 / area - 1.0)\n        - concavity (severity
of concave portions of the contour)\n        - concave points (number
of concave portions of the contour)\n        - symmetry\n        -
fractal dimension ("coastline approximation" - 1)\n\n        The mean,
standard error, and "worst" or largest (mean of the three\n
worst/largest values) of these features were computed for each image,\
n        resulting in 30 features.  For instance, field 0 is Mean
Radius, field\n        10 is Radius SE, field 20 is Worst Radius.\n\n
- class:\n                - WDBC-Malignant\n                - WDBC-
```

```
Benign\n\n    :Summary Statistics:\n\n
===================================== ====== ======\n
Min    Max\n    ===================================== ====== ======\n
radius (mean):                        6.981  28.11\n    texture
(mean):                        9.71   39.28\n    perimeter (mean):
43.79  188.5\n    area (mean):                          143.5  2501.0\
n    smoothness (mean):                    0.053  0.163\n
compactness (mean):                   0.019  0.345\n    concavity
(mean):                     0.0    0.427\n    concave points (mean):
0.0    0.201\n    symmetry (mean):                      0.106  0.304\n
fractal dimension (mean):             0.05   0.097\n    radius
(standard error):              0.112  2.873\n    texture (standard
error):              0.36   4.885\n    perimeter (standard error):
0.757  21.98\n    area (standard error):                6.802  542.2\n
smoothness (standard error):          0.002  0.031\n    compactness
(standard error):          0.002  0.135\n    concavity (standard
error):           0.0    0.396\n    concave points (standard error):
0.0    0.053\n    symmetry (standard error):            0.008  0.079\n
fractal dimension (standard error):   0.001  0.03\n    radius (worst):
7.93   36.04\n    texture (worst):                      12.02  49.54\n
perimeter (worst):                    50.41  251.2\n    area (worst):
185.2  4254.0\n    smoothness (worst):                   0.071  0.223\
n    compactness (worst):                  0.027  1.058\n    concavity
(worst):                    0.0    1.252\n    concave points (worst):
0.0    0.291\n    symmetry (worst):                     0.156  0.664\n
fractal dimension (worst):            0.055  0.208\n
===================================== ====== ======\n\n    :Missing
Attribute Values: None\n\n    :Class Distribution: 212 - Malignant,
357 - Benign\n\n    :Creator:  Dr. William H. Wolberg, W. Nick Street,
Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n    :Date: November,
1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic)
datasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a
digitized image of a fine needle\naspirate (FNA) of a breast mass.
They describe\ncharacteristics of the cell nuclei present in the
image.\n\nSeparating plane described above was obtained using\
nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\
nConstruction Via Linear Programming." Proceedings of the 4th\nMidwest
Artificial Intelligence and Cognitive Science Society,\npp. 97-101,
1992], a classification method which uses linear\nprogramming to
construct a decision tree.  Relevant features\nwere selected using an
exhaustive search in the space of 1-4\nfeatures and 1-3 separating
planes.\n\nThe actual linear program used to obtain the separating
plane\nin the 3-dimensional space is that described in:\n[K. P.
Bennett and O. L. Mangasarian: "Robust Linear\nProgramming
Discrimination of Two Linearly Inseparable Sets",\nOptimization
Methods and Software 1, 1992, 23-34].\n\nThis database is also
available through the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd
math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topic:: References\n\n
- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature
```

```
extraction \n     for breast tumor diagnosis. IS&T/SPIE 1993
International Symposium on \n     Electronic Imaging: Science and
Technology, volume 1905, pages 861-870,\n     San Jose, CA, 1993.\n
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer
diagnosis and \n     prognosis via linear programming. Operations
Research, 43(4), pages 570-577, \n     July-August 1995.\n   - W.H.
Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques\n     to diagnose breast cancer from fine-needle aspirates.
Cancer Letters 77 (1994) \n     163-171.', 'feature_names':
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal
dimension',
       'radius error', 'texture error', 'perimeter error', 'area
error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
'filename': 'breast_cancer.csv', 'data_module':
'sklearn.datasets.data'}
```

```
# loading the data to a data frame
data_frame = pd.DataFrame(breast_cancer_dataset.data, columns =
breast_cancer_dataset.feature_names)
```

```
# print the first 5 rows of the dataframe
data_frame.head()
```

```
{"type":"dataframe","variable_name":"data_frame"}
```

```
# adding the 'target' column to the data frame
data_frame['label'] = breast_cancer_dataset.target
```

```
# print last 5 rows of the dataframe
data_frame.tail()
```

```
{"type":"dataframe"}
```

```
# number of rows and columns in the dataset
data_frame.shape
```

```
(569, 31)
```

```
data_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
```

```
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
 13  area error              569 non-null    float64
 14  smoothness error        569 non-null    float64
 15  compactness error       569 non-null    float64
 16  concavity error         569 non-null    float64
 17  concave points error    569 non-null    float64
 18  symmetry error          569 non-null    float64
 19  fractal dimension error 569 non-null    float64
 20  worst radius            569 non-null    float64
 21  worst texture           569 non-null    float64
 22  worst perimeter         569 non-null    float64
 23  worst area              569 non-null    float64
 24  worst smoothness        569 non-null    float64
 25  worst compactness       569 non-null    float64
 26  worst concavity         569 non-null    float64
 27  worst concave points    569 non-null    float64
 28  worst symmetry          569 non-null    float64
 29  worst fractal dimension 569 non-null    float64
 30  label                   569 non-null    int64  
dtypes: float64(30), int64(1)
memory usage: 137.9 KB
```

```
data_frame.isnull().sum()
```

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
```

```
texture error                0
perimeter error              0
area error                   0
smoothness error             0
compactness error            0
concavity error              0
concave points error         0
symmetry error               0
fractal dimension error      0
worst radius                 0
worst texture                0
worst perimeter              0
worst area                   0
worst smoothness             0
worst compactness            0
worst concavity              0
worst concave points         0
worst symmetry               0
worst fractal dimension      0
label                        0
dtype: int64
```

```
# statistical measures about the data
data_frame.describe()
```

{"type":"dataframe"}

```
# checking the distribution of Target Varibale
data_frame['label'].value_counts()
```

```
1    357
0    212
Name: label, dtype: int64
```

1 --> Benign

0 --> Malignant

```
data_frame.groupby('label').mean()
```

{"type":"dataframe"}

Separating the features and target

```
X = data_frame.drop(columns='label', axis=1)
Y = data_frame['label']

print(X)
```

```
     mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
0          17.99         10.38          122.80     1001.0
0.11840
1          20.57         17.77          132.90     1326.0
0.08474
2          19.69         21.25          130.00     1203.0
0.10960
3          11.42         20.38           77.58      386.1
0.14250
4          20.29         14.34          135.10     1297.0
0.10030
..           ...           ...             ...        ...
...
564        21.56         22.39          142.00     1479.0
0.11100
565        20.13         28.25          131.20     1261.0
0.09780
566        16.60         28.08          108.30      858.1
0.08455
567        20.60         29.33          140.10     1265.0
0.11780
568         7.76         24.54           47.92      181.0
0.05263

     mean compactness  mean concavity  mean concave points  mean
symmetry  \
0              0.27760         0.30010              0.14710
0.2419
1              0.07864         0.08690              0.07017
0.1812
2              0.15990         0.19740              0.12790
0.2069
3              0.28390         0.24140              0.10520
0.2597
4              0.13280         0.19800              0.10430
0.1809
..                 ...             ...                  ...
...
564            0.11590         0.24390              0.13890
0.1726
565            0.10340         0.14400              0.09791
0.1752
566            0.10230         0.09251              0.05302
0.1590
567            0.27700         0.35140              0.15200
0.2397
568            0.04362         0.00000              0.00000
0.1587
```

```
     mean fractal dimension  ...  worst radius  worst texture  \
0                   0.07871  ...        25.380          17.33
1                   0.05667  ...        24.990          23.41
2                   0.05999  ...        23.570          25.53
3                   0.09744  ...        14.910          26.50
4                   0.05883  ...        22.540          16.67
..                      ...  ...           ...            ...
564                 0.05623  ...        25.450          26.40
565                 0.05533  ...        23.690          38.25
566                 0.05648  ...        18.980          34.12
567                 0.07016  ...        25.740          39.42
568                 0.05884  ...         9.456          30.37

     worst perimeter  worst area  worst smoothness  worst compactness
\
0             184.60      2019.0           0.16220            0.66560

1             158.80      1956.0           0.12380            0.18660

2             152.50      1709.0           0.14440            0.42450

3              98.87       567.7           0.20980            0.86630

4             152.20      1575.0           0.13740            0.20500

..               ...         ...               ...                ...

564           166.10      2027.0           0.14100            0.21130

565           155.00      1731.0           0.11660            0.19220

566           126.70      1124.0           0.11390            0.30940

567           184.60      1821.0           0.16500            0.86810

568            59.16       268.6           0.08996            0.06444


     worst concavity  worst concave points  worst symmetry  \
0             0.7119                0.2654          0.4601
1             0.2416                0.1860          0.2750
2             0.4504                0.2430          0.3613
3             0.6869                0.2575          0.6638
4             0.4000                0.1625          0.2364
..               ...                   ...             ...
564           0.4107                0.2216          0.2060
565           0.3215                0.1628          0.2572
566           0.3403                0.1418          0.2218
567           0.9387                0.2650          0.4087
568           0.0000                0.0000          0.2871
```

```
     worst fractal dimension
0                    0.11890
1                    0.08902
2                    0.08758
3                    0.17300
4                    0.07678
..                       ...
564                  0.07115
565                  0.06637
566                  0.07820
567                  0.12400
568                  0.07039

[569 rows x 30 columns]
```

```
print(Y)
```

```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: label, Length: 569, dtype: int64
```

Splitting the data into training data & Testing data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=2)

print(X.shape, X_train.shape, X_test.shape)
```

```
(569, 30) (455, 30) (114, 30)
```

Standardize the data

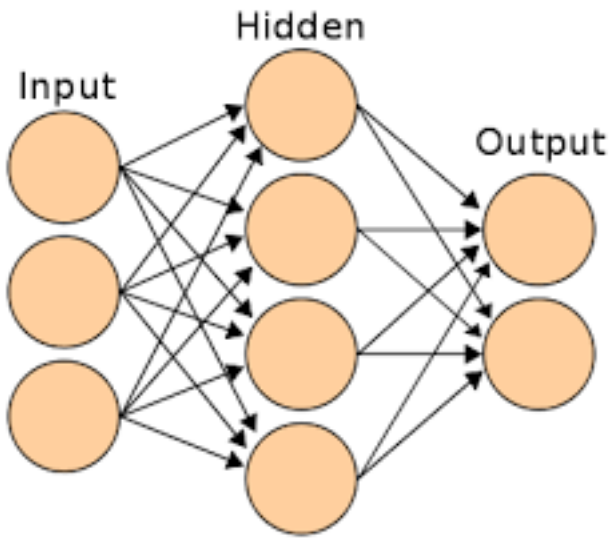
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train_std = scaler.fit_transform(X_train)

X_test_std = scaler.transform(X_test)
```

Building the Neural Network




```
# importing tensorflow and Keras
import tensorflow as tf
tf.random.set_seed(3)
from tensorflow import keras

# setting up the layers of Neural Network

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(30,)),  # Flatten layer for the
input features
    keras.layers.Dense(20, activation='relu'),  # First hidden layer
with 20 neurons and ReLU activation
    keras.layers.Dropout(0.2),  # Dropout layer for regularization
    keras.layers.Dense(40, activation='relu'),  # Second hidden layer
with 40 neurons
    keras.layers.Dropout(0.2),  # Another Dropout layer for
regularization
    keras.layers.Dense(20, activation='relu'),  # Third hidden layer
with 20 neurons
    keras.layers.Dense(2, activation='sigmoid')  # Output layer with 2
neurons for binary classification
])

# compiling the Neural Network

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# training the Neural Network
```

```
history = model.fit(X_train_std, Y_train, validation_split=0.1,
epochs=10)

Epoch 1/10
13/13 [==============================] - 1s 26ms/step - loss: 0.7072 -
accuracy: 0.5623 - val_loss: 0.6015 - val_accuracy: 0.8261
Epoch 2/10
13/13 [==============================] - 0s 5ms/step - loss: 0.5603 -
accuracy: 0.7946 - val_loss: 0.4610 - val_accuracy: 0.9348
Epoch 3/10
13/13 [==============================] - 0s 4ms/step - loss: 0.4521 -
accuracy: 0.8826 - val_loss: 0.3313 - val_accuracy: 0.9565
Epoch 4/10
13/13 [==============================] - 0s 4ms/step - loss: 0.3372 -
accuracy: 0.9071 - val_loss: 0.2338 - val_accuracy: 0.9565
Epoch 5/10
13/13 [==============================] - 0s 6ms/step - loss: 0.2402 -
accuracy: 0.9242 - val_loss: 0.1751 - val_accuracy: 0.9565
Epoch 6/10
13/13 [==============================] - 0s 6ms/step - loss: 0.1892 -
accuracy: 0.9438 - val_loss: 0.1432 - val_accuracy: 0.9565
Epoch 7/10
13/13 [==============================] - 0s 5ms/step - loss: 0.1765 -
accuracy: 0.9340 - val_loss: 0.1258 - val_accuracy: 0.9565
Epoch 8/10
13/13 [==============================] - 0s 5ms/step - loss: 0.1480 -
accuracy: 0.9487 - val_loss: 0.1143 - val_accuracy: 0.9565
Epoch 9/10
13/13 [==============================] - 0s 8ms/step - loss: 0.1197 -
accuracy: 0.9535 - val_loss: 0.1009 - val_accuracy: 0.9565
Epoch 10/10
13/13 [==============================] - 0s 8ms/step - loss: 0.1120 -
accuracy: 0.9682 - val_loss: 0.0909 - val_accuracy: 0.9565
```

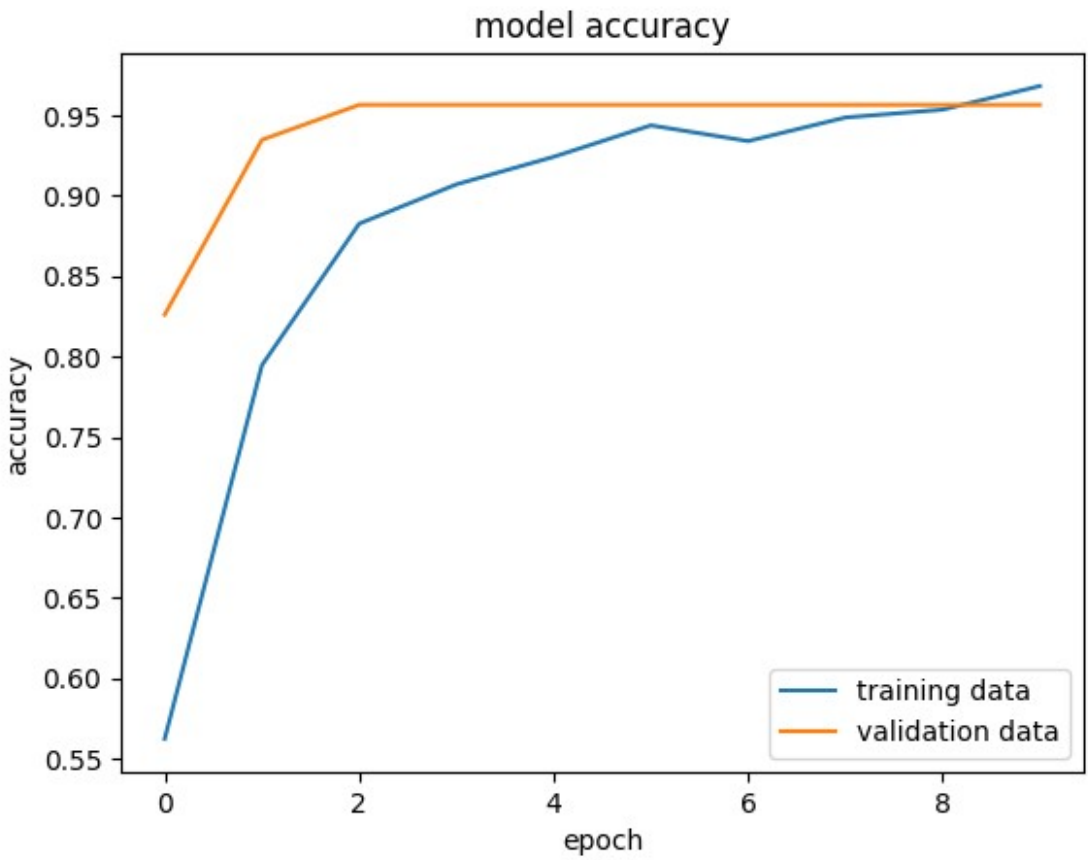
Visualizing accuracy and loss

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')

plt.legend(['training data', 'validation data'], loc = 'lower right')

<matplotlib.legend.Legend at 0x7d41c44b37c0>
```

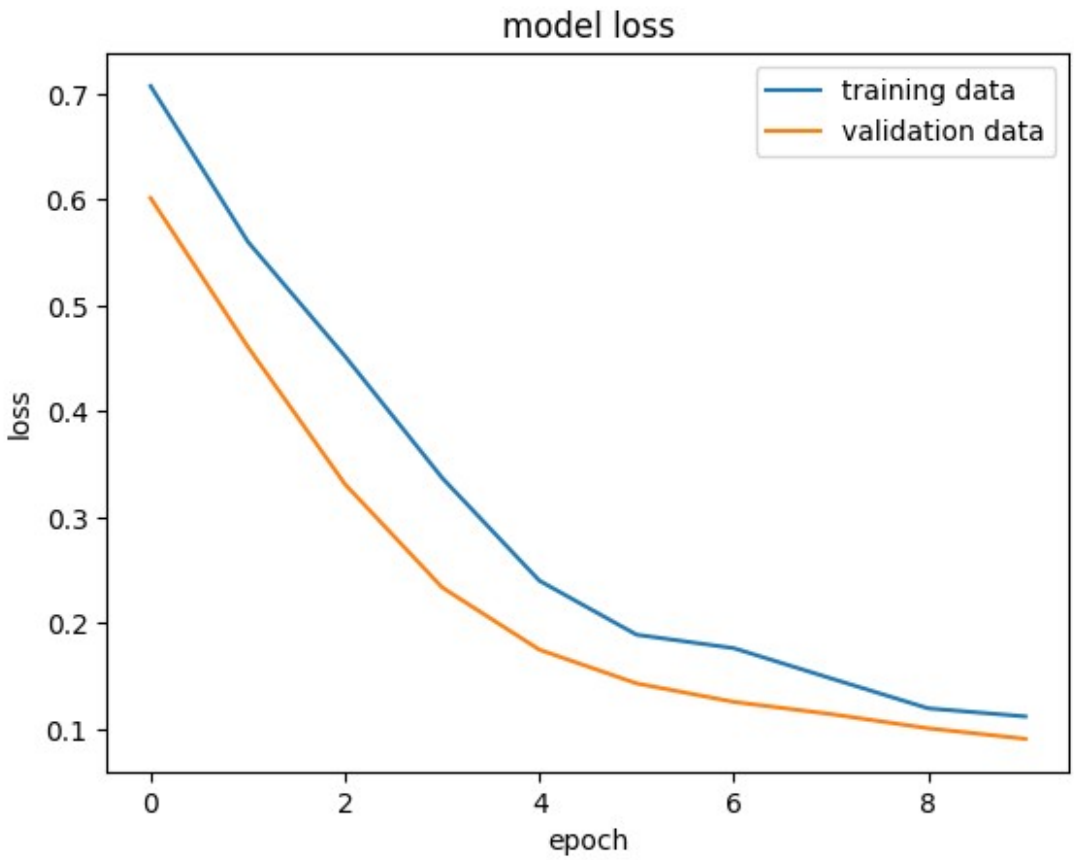



```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')

plt.legend(['training data', 'validation data'], loc = 'upper right')
```

```
<matplotlib.legend.Legend at 0x7d41d4eac220>
```



Accuracy of the model on test data

```
loss, accuracy = model.evaluate(X_test_std, Y_test)
print(accuracy)
```

```
4/4 [==============================] - 0s 4ms/step - loss: 0.1360 -
accuracy: 0.9386
0.9385964870452881
```

```
print(X_test_std.shape)
print(X_test_std[0])
```

```
(114, 30)
[-0.04462793 -1.41612656 -0.05903514 -0.16234067  2.0202457  -
0.11323672
  0.18500609  0.47102419  0.63336386  0.26335737  0.53209124
2.62763999
  0.62351167  0.11405261  1.01246781  0.41126289  0.63848593
2.88971815
 -0.41675911  0.74270853 -0.32983699 -1.67435595 -0.36854552 -
0.38767294
  0.32655007 -0.74858917 -0.54689089 -0.18278004 -1.23064515 -
0.6268286 ]
```

```
Y_pred = model.predict(X_test_std)
```

```
4/4 [==============================] - 0s 3ms/step
```

```
print(Y_pred.shape)
print(Y_pred[0])
```

```
(114, 2)
[0.4761347 0.5891135]
```

```
print(X_test_std)
```

```
[[-0.04462793 -1.41612656 -0.05903514 ... -0.18278004 -1.23064515
  -0.6268286 ]
 [ 0.24583601 -0.06219797  0.21802678 ...  0.54129749  0.11047691
   0.0483572 ]
 [-1.26115925 -0.29051645 -1.26499659 ... -1.35138617  0.269338
  -0.28231213]
 ...
 [ 0.72709489  0.45836817  0.75277276 ...  1.46701686  1.19909344
   0.65319961]
 [ 0.25437907  1.33054477  0.15659489 ... -1.29043534 -2.22561725
  -1.59557344]
 [ 0.84100232 -0.06676434  0.8929529  ...  2.15137705  0.35629355
   0.37459546]]
```

```
print(Y_pred)
```

```
[[0.4761347  0.5891135 ]
 [0.64365935 0.33962882]
 [0.05444277 0.90630305]
 [0.99997294 0.00395588]
 [0.40848053 0.6482865 ]
 [0.9989118  0.0159086 ]
 [0.26139945 0.6786583 ]
 [0.01705254 0.9620426 ]
 [0.0613014  0.90634596]
 [0.0875355  0.90893847]
 [0.5124762  0.4694274 ]
 [0.12992992 0.8330656 ]
 [0.22269225 0.7353055 ]
 [0.2192408  0.7609455 ]
 [0.09464807 0.8775621 ]
 [0.98872876 0.14006248]
 [0.0730932  0.91577214]
 [0.10614242 0.83984613]
 [0.10858931 0.84064674]
 [0.9965368  0.02954681]
 [0.21509483 0.991838  ]
 [0.03645335 0.9127266 ]
 [0.06206032 0.88285565]
```

```
[0.03356086 0.95445484]
[0.21577021 0.78122914]
[0.9909987  0.04969655]
[0.11825524 0.8257079 ]
[0.1958513  0.7977523 ]
[0.9932951  0.03833378]
[0.9902719  0.06376764]
[0.09230069 0.9188829 ]
[0.07428423 0.8799522 ]
[0.07914666 0.9119148 ]
[0.99991375 0.00308872]
[0.9930998  0.02749448]
[0.17574221 0.7379421 ]
[0.01119537 0.9665802 ]
[0.10420641 0.8383291 ]
[0.04189317 0.92571294]
[0.07684108 0.89777136]
[0.99986    0.00371659]
[0.9066006  0.19624409]
[0.022027   0.9229008 ]
[0.10490183 0.86065996]
[0.8954169  0.17827019]
[0.08126944 0.9119304 ]
[0.01880276 0.9807071 ]
[0.12010784 0.831408  ]
[0.99981785 0.00779324]
[0.96501166 0.1031717 ]
[0.06882574 0.93282455]
[0.89837563 0.14445534]
[0.6305418  0.39654404]
[0.05301078 0.90843934]
[0.04527919 0.9271272 ]
[0.45117474 0.60768586]
[0.1090745  0.80946714]
[0.03462956 0.93163025]
[0.97360027 0.14533041]
[0.07433323 0.90738964]
[0.2240819  0.75442404]
[0.9842953  0.11707667]
[0.02812734 0.9501039 ]
[0.97299963 0.07900458]
[0.9804809  0.05901429]
[0.37792602 0.91015196]
[0.9980361  0.02800795]
[0.9769356  0.0884546 ]
[0.508042   0.4825158 ]
[0.5371423  0.5463292 ]
[0.92997855 0.26834506]
[0.99672824 0.04472074]
```

```
[0.08401683 0.85219723]
[0.8474645  0.22978401]
[0.01537887 0.96810853]
[0.85383976 0.21114857]
[0.09270663 0.85222715]
[0.0421484  0.9651338 ]
[0.25887913 0.6877607 ]
[0.78209585 0.30550388]
[0.9793985  0.12025438]
[0.83651584 0.30756515]
[0.966492   0.07298633]
[0.21370885 0.78099394]
[0.14968595 0.81808364]
[0.81426835 0.1989889 ]
[0.07568808 0.90806335]
[0.10284016 0.90589076]
[0.13891609 0.84774274]
[0.9981383  0.03741442]
[0.04869932 0.92071337]
[0.08956094 0.8707655 ]
[0.0487     0.9248534 ]
[0.9946757  0.08961872]
[0.86687946 0.18211856]
[0.07346562 0.9263759 ]
[0.992028   0.04296107]
[0.95461494 0.09815646]
[0.1040169  0.855782  ]
[0.02584049 0.9532298 ]
[0.01785903 0.9694233 ]
[0.8963717  0.20508191]
[0.9998633  0.01080173]
[0.99929494 0.0098044 ]
[0.07377093 0.87473136]
[0.05562583 0.93240607]
[0.03346116 0.9245121 ]
[0.03472186 0.92667514]
[0.03699816 0.9124247 ]
[0.16355745 0.77345926]
[0.9977181  0.02303088]
[0.9933765  0.040245  ]
[0.86221534 0.26522905]
[0.9802137  0.08451164]]
```

model.predict() gives the prediction probability of each class for that data point

```
#  argmax function

my_list = [0.25, 0.56]
```

```
index_of_max_value = np.argmax(my_list)
print(my_list)
print(index_of_max_value)
```

```
[0.25, 0.56]
1
```

```
# converting the prediction probability to class labels

Y_pred_labels = [np.argmax(i) for i in Y_pred]
print(Y_pred_labels)
```

```
[1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0,
1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1,
1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
```

Building the predictive system

```
input_data =
(11.76,21.6,74.72,427.9,0.08637,0.04966,0.01657,0.01115,0.1495,0.05888
,0.4062,1.21,2.635,28.47,0.005857,0.009758,0.01168,0.007445,0.02406,0.
001769,12.98,25.72,82.98,516.5,0.1085,0.08615,0.05523,0.03715,0.2433,0
.06563)

# change the input_data to a numpy array
input_data_as_numpy_array = np.asarray(input_data)

# reshape the numpy array as we are predicting for one data point
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

# standardizing the input data
input_data_std = scaler.transform(input_data_reshaped)

prediction = model.predict(input_data_std)
print(prediction)

prediction_label = [np.argmax(prediction)]
print(prediction_label)

if(prediction_label[0] == 0):
  print('The tumor is Malignant')

else:
  print('The tumor is Benign')
```

```
1/1 [==============================] - 0s 21ms/step
[[0.04607509 0.90112996]]
[1]
The tumor is Benign
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439:
UserWarning: X does not have valid feature names, but StandardScaler
was fitted with feature names
  warnings.warn(
```